Università degli Studi di Genova

Dipartimento di Matematica

Corso di studi in

Statistica Matematica
e trattamento Informatico dei Dati

Anno accademico 2023/2024

Tesi di Laurea

Reti Neurali Convoluzionali ed approccio moderno alla Computer Vision

Candidato
Gualtiero Marenco Turi

Relatore interno
Prof. Ennio Ottaviani

Relatore esterno
Ing. Andrea Rapuzzi

Indice

Capitolo 1

Introduzione

1.1 Reti neurali artificiali

Nel contesto del Machine Learning, le reti neurali artificiali (ANN) costituiscono una categoria di modelli di apprendimento statistico ispirati alle configurazioni neurali presenti nei cervelli di diversi mammiferi. Tali reti sono impiegate per l'approssimazione o l'analisi di funzioni definite da un ampio insieme di dati in ingresso, solitamente di natura non conosciuta. La rappresentazione delle reti neurali artificiali avviene comunemente mediante sistemi di "neuroni" interconnessi, instaurando un flusso di comunicazione. Ciascuna connessione all'interno di tali reti è associata a un peso numerico, modificabile in base all'esperienza acquisita, conferendo alle reti neurali la capacità di adattarsi dinamicamente agli input e di simulare la capacità umana di apprendimento.

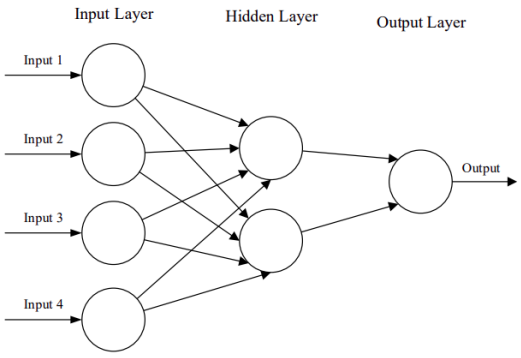

Figura 1.1: Struttura esemplificativa di una semplice ANN

La struttura di base di una ANN può essere modellata come mostrato nella figura 1.1. L'input, solitamente sotto forma di un vettore multidimensionale, viene caricato nel layer di input (strato di input) che in seguito lo distribuisce agli hidden layers, ovvero gli strati intermedi di neuroni tra l'input layer e l'output layer (strato di output). Gli hidden layer prendono poi le elaborazioni dal layer

precedente e valutano come un cambiamento stocastico al loro interno deteriori o migliori l'output finale, ripetendo poi il passaggio e definendo così il processo di apprendimento. Avere molteplici hidden layer concatenati uno dopo l'altro è ciò che viene comunemente chiamato deep learning.
Come illustrato nei prossimi capitoli, uno dei campi di applicazione più comune per alcuni tipi di ANN è la classificazione di immagini. La prima fondamentale distinzione quando si tratta di modelli di elaborazione delle immagini è quella tra l'apprendimento supervisionato e l'apprendimento non supervisionato. L'apprendimento supervisionato avviene attraverso input pre-etichettati (labeled), che agiscono come obbiettivi. Per ogni campione di addestramento vi è un insieme di valori di input (vettori) e uno o più valori di output designati associati. L'apprendimento non supervisionato differisce in quanto il set di addestramento non include etichette. Nel caso supervisionato il successo è solitamente determinato dalla capacità della rete di ridurre o aumentare una funzione di costo associata, comunemente chiamata Loss Function. Generalmente, la maggior parte dei compiti di riconoscimento di pattern focalizzati sull'immagine si concentra sulla classificazione utilizzando l'apprendimento supervisionato.

1.2 Reti neurali convoluzionali

Le reti neurali convoluzionali (CNN) sono analoghe alle tradizionali reti neurali artificiali (ANN): sono anch'esse composte da neuroni che si auto-ottimizzano attraverso l'apprendimento ed ogni neurone riceverà anche in questo caso un input ed eseguirà un'operazione (un prodotto scalare seguito da una funzione non lineare), come avviene di consueto in ogni ANN. La fondamentale differenza tra le reti neurali convoluzionali (CNN) e le reti neurali artificiali tradizionali (ANN) sta nella specializzazione: le CNN sono prettamente utilizzate nel campo del riconoscimento di pattern all'interno delle immagini e ciò consente di implementare caratteristiche specifiche nell'architettura, ottimizzandole per compiti focalizzati, mentre si riducono ulteriormente i parametri necessari per configurare il modello. Nelle reti neurali convoluzionali, durante l'esecuzione delle operazioni di convoluzione, i pesi del kernel (o filtro) vengono condivisi tra tutti i nodi dello strato di output. Tale condivisione implica che, durante il processo di convoluzione, il medesimo kernel venga applicato iterativamente a differenti sezioni dell'immagine di input. Questo meccanismo di condivisione dei pesi permette alle CNN di apprendere in modo efficiente i pattern locali presenti nell'immagine, concomitantemente riducendo il numero complessivo dei parametri soggetti ad addestramento. Una delle limitazioni più grandi delle forme tradizionali di ANN infatti è che tendono ad avere difficoltà con la complessità computazionale richiesta per elaborare dati di immagine. Comuni set di dati di riferimento per il machine learning, come il database MNIST (modified National Institute of Standards and Technology database) di cifre scritte a mano, sono adatti per la maggior parte delle forme di ANN,

solo grazie alla loro dimensione relativamente ridotta delle immagini, di soli 28 × 28 pixel nel caso di MNIST. Con questo dataset, un singolo neurone nel primo strato nascosto conterrà 784 pesi (28×28×1, tenendo presente che MNIST è limitato solamente ai colori bianco e nero) che è gestibile per la maggior parte delle forme di ANN. Tuttavia, se invece si considera un input a colori o più grande, ad esempio 64 × 64 pixel, il numero di pesi su un singolo neurone del primo strato aumenta notevolmente a 12.288. Bisogna tenere conto dunque che per gestire questa scala di input la rete dovrà essere molto più estesa di quella utilizzata per classificare cifre MNIST. Tuttavia, semplicemente addestrare una ANN di dimensioni maggiori non sempre è una soluzione attuabile. Ciò è dovuto a due motivi, in primis il semplice problema di non avere abbastanza potenza computazionale e tempo per addestrare una rete neurale artificiale grande a piacere. La seconda ragione è evitare gli effetti dell'overfitting, che si verifica quando il risultato di un modello corrisponde troppo strettamente, se non esattamente, a un particolare insieme di dati, e può quindi non adattarsi a dati aggiuntivi o prevedere osservazioni future attendibilmente. Quanto appena descritto è un concetto chiave per la maggior parte degli algoritmi di apprendimento automatico ed è essenziale prendere le precauzioni necessarie a ridurne gli effetti. Se i modelli mostrassero segni di overfitting, si assisterebbe ad una ridotta capacità di individuare caratteristiche generalizzate non solo nel dataset di addestramento in questione, ma anche in quelli di test e previsione. L'overfitting è dunque il motivo principale per cui si mira a ridurre la complessità delle ANN: meno parametri sono necessari per l'addestramento, meno probabilità ci sono che la rete ne venga negativamente influenzata. Una delle distinzioni fondamentali tra una generica ANN e una CNN risiede nel fatto che i neuroni, all'interno degli strati di una CNN, sono organizzati in tre dimensioni: la dimensione spaziale dell'input (altezza e larghezza) e la profondità. La profondità, in questo contesto, non si riferisce al numero totale di strati presenti in una ANN, bensì a una terza dimensione di un volume di attivazione. In contrasto con le ANN convenzionali, i neuroni all'interno di uno specifico strato si collegano esclusivamente a una ristretta regione del precedente strato. Concretamente, ciò implica che, nel caso dell'esempio precedentemente menzionato, il 'volume' di input avrà una dimensione di 64 × 64 × 3 (altezza, larghezza e profondità). Questo porta a uno strato di output finale caratterizzato da una dimensionalità di 1 × 1 × n, dove n rappresenta il potenziale numero di classi. Tale risultato è ottenuto condensando l'intera dimensionalità di input in un volume più compatto di punteggi di classe distribuiti lungo la dimensione della profondità.

Capitolo 2

Architettura di una CNN

2.1 L'input di immagini

Un'immagine digitale può essere definita come la rappresentazione numerica di un'immagine reale. Tale rappresentazione può essere codificata come un vettore o una matrice detta bitmap (o raster). Nelle immagini bitmap a colori, il colore è memorizzato come intensità dei colori di base, ad esempio nel modello RGB, che comprende tre colori principali: rosso, verde e blu. Nelle immagini bitmap in scala di grigi (spesso impropriamente definite in bianco e nero), il valore indica varie intensità di grigio che vanno dal nero al bianco. La profondità di colore o di grigio possibili dipende dalla quantità di bit utilizzati per codificarli: ad esempio, un'immagine con 1 bit per pixel può rappresentare solo due colori (0 e 1), mentre un'immagine con 8 bit per pixel può rappresentare 256 colori o livelli di grigio, e così via. Le immagini bitmap possono essere memorizzate in diversi formati, spesso basati su algoritmi di compressione. Tali algoritmi possono essere lossy (come JPEG) o lossless (senza perdita, come GIF o PNG). Questo tipo di immagini è generato da una vasta gamma di dispositivi di acquisizione, come scanner, fotocamere digitali, satelliti, webcam, radar e microscopi elettronici.

2.2 Gli strati di una CNN

Le CNN sono costituite da tre categorie di strati: strati convoluzionali, strati di pooling e strati completamente connessi. L'interconnessione di questi strati dà vita a un'architettura CNN. Una rappresentazione semplificata di un'architettura CNN è esemplificata nella figura 2.1. Una CNN riceve in ingresso un insieme di p immagini rappresentate come singole bitmap in scala di grigi (ad esempio, i tre canali R, G e B di un'immagine a colori). Tali immagini vengono quindi elaborate da un modulo di estrazione delle caratteristiche basato su operatori di convoluzione che produce un array di m elementi, corrispondente alle caratteristiche estratte. Questo array viene infine sottoposto a una rete neurale completamente connessa

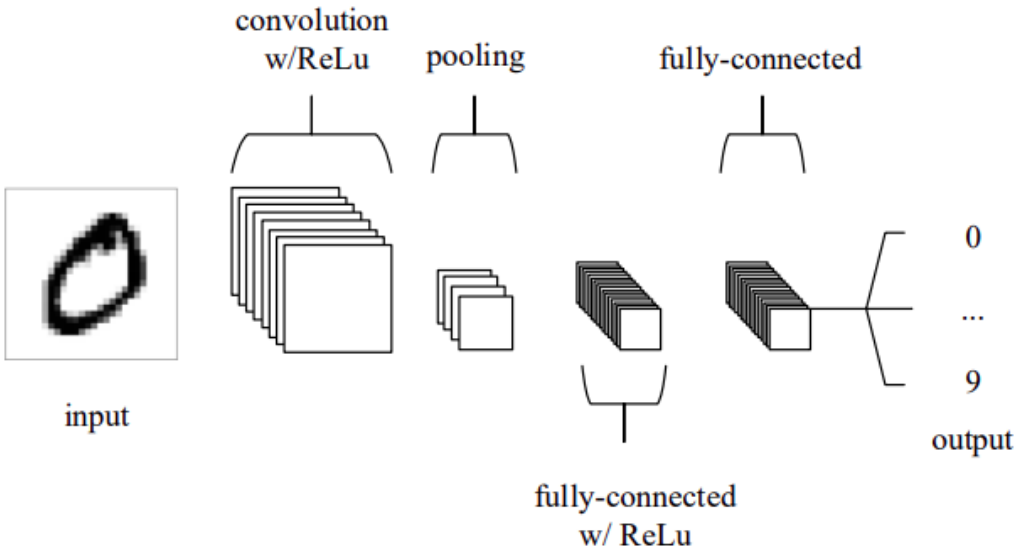

Figura 2.1: Una semplice CNN con un solo layer convoluzionale e un layer di sottocampionamento.

per generare i risultati desiderati. Tale rete neurale è composta tipicamente da n percettroni sia in ingresso che in uscita, dove n rappresenta il numero di classi da classificare, e può anche essere essa stessa multi-strato. Il modulo di estrazione delle caratteristiche, solitamente, è costituito da uno strato convoluzionale seguito da zero, uno o più coppie di strati di sottocampionamento e strati convoluzionali. Pertanto, tale modulo presenta un numero dispari di sottomoduli, dove lo strato convoluzionale rappresenta il primo e l'ultimo elemento. Va notato che sebbene per via della natura dell'operazione di convoluzione più applicazioni di quest'ultima possano essere sempre condensate in una sola cambiando opportunamente il filtro, l'esistenza della funzione di attivazione che introduce non-linearità in ogni strato convoluzionale permette di combinare efficacemente più blocchi. L'array di caratteristiche finale è il risultato dell'ultimo strato convoluzionale, in cui le immagini di input sono trasformate in singoli valori. Se k rappresenta il numero di strati all'interno della CNN, questa può essere denominata una k-CNN, o più semplicemente, una CNN con k strati. In linea di principio, un aumento di k corrisponde a un incremento della capacità della rete di apprendere pattern complessi. Tuttavia, nella pratica, un eccessivo numero di strati può causare problemi di overfitting durante il processo di addestramento, a seconda della quantità di dati disponibili.
Di seguito è stata fornita una analisi su una generale rete neurale convoluzionale. Tuttavia, esistono molte varianti a seconda del compito assegnato: per esempio, esistono applicazioni di CNN senza la rete neurale finale. Infine, per l'addestramento di una CNN, viene impiegato l'approccio classico supervisionato basato

sulla retropropagazione (backpropagation). Questo metodo consente alla rete di ottimizzare i pesi condivisi degli strati convolutivi basandosi sul set di addestramento. I seguenti paragrafi illustrano una spiegazione accurata del funzionamento e della struttura dei layers di una CNN.

2.3 Convolution layer (Strato di convoluzione)

2.3.1 Filtro o Kernel

Il cuore di una CNN è il filtro o kernel, in verde nell'immagine 2.2, che è una piccola matrice di pesi. Questo filtro scorre sull'input dell'immagine attraverso una finestra di passi (stride), e i prodotti puntuali vengono sommati per creare la mappa delle caratteristiche (feature map).

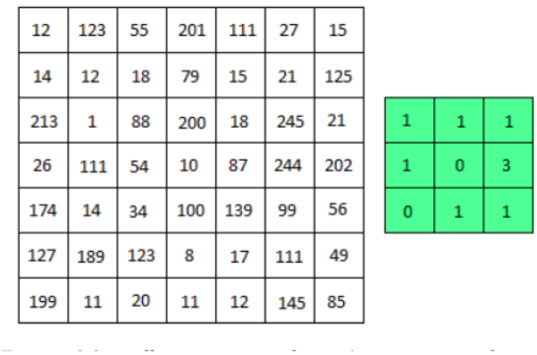

Figura 2.2: Esempio di matrice dei dati e filtro

2.3.2 L'operazione di convoluzione

La convoluzione è un'operazione eseguita su un'immagine per enfatizzare alcune delle sue caratteristiche. Per fare ciò, è necessario un'immagine digitale e una matrice di convoluzione, anche nota come filtro. Questo ultimo può essere considerato come una finestra scorrevole che si muove lungo l'immagine originale. Ad ogni spostamento, esso produce un nuovo valore ottenuto sommando tutti i prodotti tra gli elementi del filtro e i pixel corrispondenti. I valori ottenuti da tutte le posizioni possibili del filtro sull'immagine vengono ordinatamente inseriti in una nuova immagine.

L'operazione di convoluzione genera un'immagine che evidenzia le caratteristiche enfatizzate dal filtro utilizzato. Pertanto, per quanto riguarda le dimensioni della nuova immagine, si applica la seguente formula:

$$b_n = b_v - b_f + 1 \quad \text{e} \quad h_n = h_v - h_f + 1$$

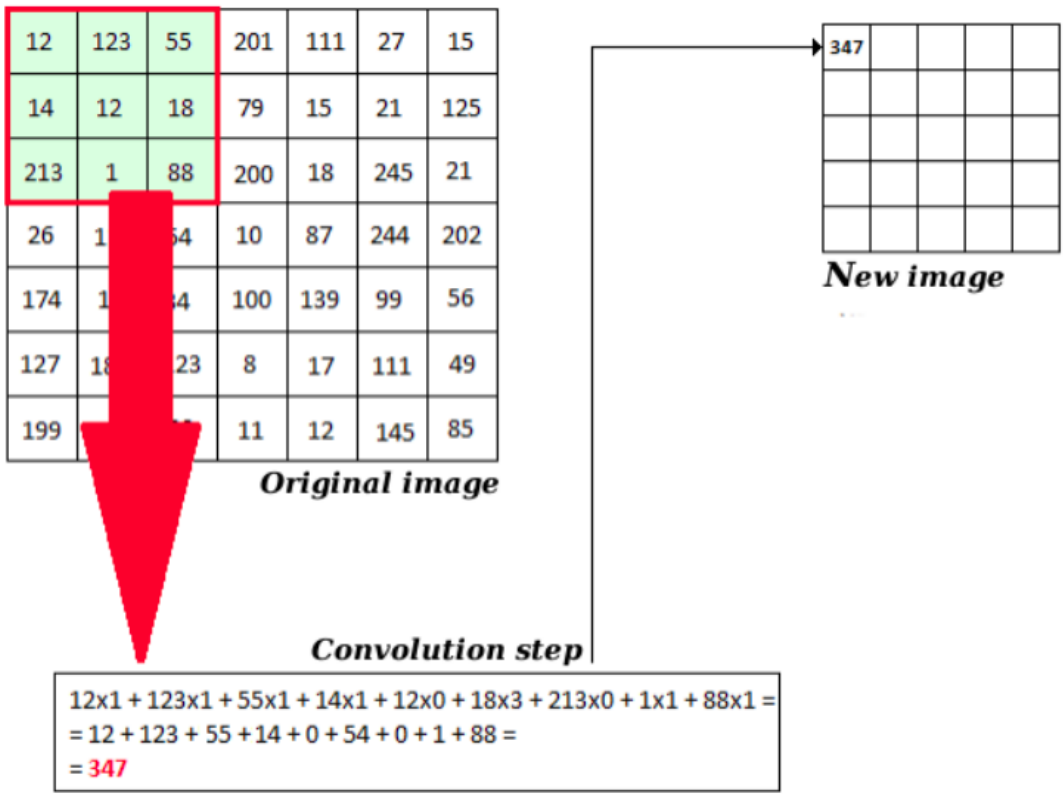

Figura 2.3: Convoluzione uno a uno

dove:

- b_n e h_n rappresentano rispettivamente la larghezza e l'altezza della nuova immagine risultante dalla convoluzione
- b_v e h_v rappresentano rispettivamente la larghezza e l'altezza dell'immagine originale;
- b_f e h_f rappresentano rispettivamente la larghezza e l'altezza del filtro utilizzato, tipicamente 3x3.

Nell'elaborazione delle immagini vengono impiegati diversi filtri, ognuno con uno scopo specifico. Alcuni dei più diffusi includono:

- Filtro high-pass: per incrementare i dettagli dell'immagine;
- Filtri di Gauss: adoperati per rimuovere il rumore;
- Filtri di Sobel: utilizzati per evidenziare i bordi;
- Filtro in rilievo: per accentuare le differenze di luminosità.

L'operazione di convoluzione rappresenta un elemento cruciale nelle reti neurali convoluzionali, in cui i filtri vengono appresi automaticamente durante la fase di addestramento, senza supervisione. Questo aspetto costituisce la reale innovazione introdotta dalle CNN nella computer vision classica, in cui gli operatori di convoluzione (detti filtri) venivano progettati da esperti sulla base della conoscenza del problema specifico da trattare. Si ottenevano buoni risultati, ma non era poi possibile trasferire l'esperienza fatta su nuovi problemi. L'avvento delle CNN ha consentito di rendere automatica (ed ottimale) la stima dei filtri da applicare sulle immagini per produrre le features che servono a classificare.

Fino ad ora sono state trattate esclusivamente le convoluzioni uno-a-uno, le quali ricevono un'unica immagine in input e producono un'unica immagine in output. Tuttavia, esistono diverse altre tipologie di convoluzione ampiamente impiegate nelle reti neurali convoluzionali (Immagine 2.4), tra esse vanno citate le convoluzioni uno-a-molti, in cui viene fornita un'unica immagine in input e vengono applicati più filtri. Inoltre, esistono le convoluzioni molti-a-molti, che coinvolgono più immagini sia di input sia di output. Ogni immagine di output può essere collegata con una o più immagini di input. Ciascuna di queste connessioni tra un'immagine di input e un'immagine di output è caratterizzata da un filtro. Per ogni pixel dell'immagine di output, viene prima eseguita la corrispondente operazione di convoluzione e successivamente i risultati intermedi vengono sommati insieme. È importante notare che all'aumentare del numero totale di connessioni, le immagini vengono elaborate in modo più approfondito durante il processo di apprendimento. Tuttavia, è altrettanto evidente che un tale incremento delle connessioni comporta un aumento significativo del tempo computazionale.

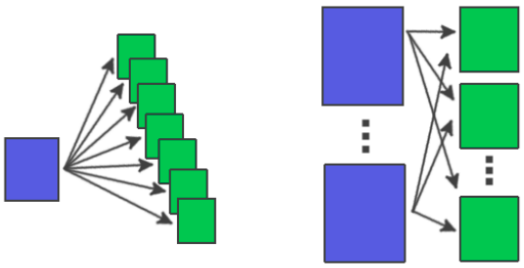

Figura 2.4: Convoluzione una a tanti e tanti a tanti

2.3.3 Zero padding

Spesso, il bordo dell'immagine viene "riempito" con zeri per garantire che l'output della convoluzione abbia le stesse dimensioni dell'input. Questa procedura è chiamata padding e aiuta a mantenere le informazioni lungo i bordi dell'immagine. In alternativa, l'output risulta di dimensione ridotte sulla base della dimensione del kernel utilizzato.

2.3.4 Funzione di attivazione

Dopo la convoluzione, viene applicata una funzione di attivazione, tipicamente ReLU (Rectified Linear Unit), che aggiunge non linearità all'output introducendo la componente di attivazione. Ogni neurone, dopo aver ricevuto input pesati, li somma e applica la funzione di attivazione al risultato. L'output della funzione di attivazione diventa quindi l'input per il successivo strato della rete. Le funzioni di attivazione servono a introdurre non linearità nelle reti neurali, rendendo possibile per la rete apprendere e rappresentare modelli complessi, anche in presenza di relazioni non lineari nei dati. Senza funzioni di attivazione o con funzioni di attivazione lineari, la rete sarebbe in grado di approssimare solo funzioni lineari, limitando notevolmente la sua capacità di apprendimento. Alcune delle funzioni di attivazione più comuni includono:

- Sigmoid:
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$
- Tangente iperbolica (tanh):
$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1}$$
- ReLU (Rectified Linear Unit):
$$f(x) = \max(0, x)$$
- Leaky ReLU:
$$f(x) = \max(\alpha x, x) \quad \text{dove } \alpha \text{ è un piccolo valore costante.}$$
- Softmax: Utilizzata solitamente nell'ultimo strato di una rete per problemi di classificazione multiclasse. La formula della funzione Softmax per un vettore di valori $\mathbf{z}$ di lunghezza K è definita come:
$$\text{softmax}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$
dove $\text{softmax}(\mathbf{z})_i$ rappresenta l'elemento i-esimo del vettore trasformato mediante Softmax.

2.4 Downsampling Layer (Strato di Sottocampionamento)

Nelle reti neurali convoluzionali, riveste rilevanza un altro modulo cruciale denominato strato di sottocampionamento. La sua funzione primaria è quella di ridurre le dimensioni delle immagini in ingresso al fine di conferire all'algoritmo maggiore invarianza e robustezza. Lo strato di sottocampionamento accetta n immagini di input e restituisce n output; le immagini da sottocampionare vengono divise in blocchi tutti di identiche dimensioni, dove ciascun blocco è quindi mappato su un singolo pixel come segue:

$$y = f\left(\sum_{\forall(i,j)\in B} B_{i,j} \cdot w + b\right)$$

dove:

- y denota il risultato del passaggio di sottocampionamento;
- f rappresenta la funzione di attivazione;
- B indica il blocco considerato nell'immagine di input;
- $B_{i,j}$ rappresenta il valore del pixel (i, j) all'interno del blocco B;
- w rappresenta il coefficiente di regolazione;
- b rappresenta la soglia.

Il calcolo eseguito su un'immagine di input da uno strato di sottocampionamento è condotto tramite un percettrone dotato di pesi uniformi (eccetto per la soglia), pertanto, l'addestramento di n percettroni è sufficiente per generare 2n parametri. E' da notare che uno strato di sottocampionamento risulta meno potente di uno strato convoluzionale in quanto, come precedentemente accennato, una maggiore quantità di parametri addestrabili aumenta la capacità discrezionale della rete. Vi sono tre motivi principali per cui tuttavia non si utilizzano esclusivamente strati convoluzionali:

- Una minore quantità di parametri rende il processo di addestramento più rapido:
- Uno strato di sottocampionamento esegue le sue operazioni con maggiore velocità rispetto a uno strato convoluzionale grazie a un minor numero di passaggi e prodotti;

- Lo strato di sottocampionamento permette alle CNN di sopportare traslazioni e rotazioni tra i modelli di input. A tal fine, uno strato di sottocampionamento singolo non risulta sufficiente, al contrario, è stato osservato che un'alternanza di estrazione delle caratteristiche e strati di sottocampionamento può gestire una maggior varietà di input che una sequenza di soli strati convoluzionali non potrebbe trattare.

- È infine possibile personalizzare uno strato di sottocampionamento in modo da suddividere le immagini di input in blocchi sovrapposti. Questa configurazione rallenterebbe notevolmente il processo di riduzione, il quale potrebbe, a seconda del compito, richiedere tempi di esecuzione estremamente ristretti sia nella fase di addestramento che in quella di test.

Talvolta, è possibile impiegare anche strati di sottocampionamento più semplici per accelerare il processo di addestramento. Un esempio eccellente di questo approccio è rappresentato dal cosiddetto strato di max-pooling, dove vengono eseguite esclusivamente le seguenti operazioni:

- A viene trovato prendendo il valore max tra i pixel nel blocco;

- Viene applicata la funzione di attivazione su A.

In una CNN, ogni strato di sottocampionamento può essere sostituito con uno strato di max-pooling, caratterizzato da una minore quantità di parametri da addestrare grazie all'eliminazione dei coefficienti di regolazione. Ciò si traduce in una maggiore velocità durante la fase di addestramento, spesso a scapito di un livello inferiore di apprendimento.

2.5 Fully Connected Layer (Strato completamente connesso)

Le ultime fasi di una CNN coinvolgono strati completamente connessi, simili a quelli nelle reti neurali tradizionali di tipo MLP (Multi-Layer Perceptron). Questi strati convertono le caratteristiche estratte dalle fasi precedenti in una forma adatta a seconda del compito assegnato.

2.6 Training

Le CNN sono storicamente addestrate tramite la tecnica della back-propagation (retropropagazione). La tecnica di back-propagation richiede di considerare, per ogni iterazione, un nuovo modello di training x. Supponendo che l'output dello strato nascosto sia $q_1, q_2, q_3, \ldots, q_m$, è possibile effettuare i seguenti calcoli:

$$q_i = f(h_i x) \quad \forall i = 1, \ldots, m$$

Successivamente, $y_1, y_2, y_3, \ldots, y_n$, l'output del MLP, può essere calcolato come segue:

$$y_i = f(w_i^T q) \quad \forall i = 1, \ldots, n$$

Dove:

- w_i è il vettore dei pesi associato all'i-esimo percettrone nello strato di output;
- h_i è il vettore dei pesi associato all'i-esimo percettrone nello strato nascosto;
- f è la funzione di attivazione.

L'errore quadratico medio finale su x del MLP può essere calcolato come segue:

$$T_x = \frac{1}{2} \sum_{i=1}^{n} (z_i - y_i)^2$$

Nell'equazione precedente, z_i è il risultato esatto che l'i-esimo percettrone di output dovrebbe produrre e che può essere recuperato dal set di training. Come per il percettrone, si desidera minimizzare l'errore trovando la giusta combinazione di valori per w_i per $i = 1, \ldots, n$ e h_j per $j = 1, \ldots, m$. Anche in questo caso, una tecnica di discesa del gradiente può essere applicata per minimizzare la funzione di errore. Per la minimizzazione iniziamo calcolando l'errore dallo strato di output:

$$e_i = (z_i - y_i) f'(w_i^T q) \quad \forall i \in \{1, \ldots, n\}$$

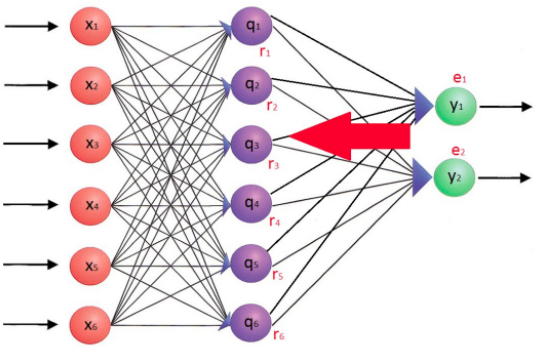

Figura 2.5: Ad ogni iterazione un errore è associato ad ogni percettrone per aggiornare gli pesi di conseguenza

Poi l'errore può essere propagato all'indietro, basandosi su quello calcolato sullo strato di output:

$$r_i = f'(h_i x) \sum_{k=i}^{n} e_k w_{ki} \quad \forall i \in \{1, \ldots, m\}$$

In questo caso stiamo considerando una rete neurale con un solo strato nascosto, ma la back-propagation può essere facilmente estesa a qualsiasi tipo di rete ripetendo il calcolo mostrato precedentemente trattando lo strato nascosto successivo come se fosse lo strato di output. Dopo la back-propagation dell'errore, i pesi di input dei percettroni di output possono essere aggiornati:

$$w_i = w_i + \alpha \cdot e_i \cdot q \quad \forall i \in \{1, \ldots, n\}$$

E i pesi di input dello strato nascosto possono essere adattati come segue:

$$h_i = h_i + \alpha \cdot r_i \cdot x \quad \forall i \in \{1, \ldots, m\}$$

Con le ultime due equazioni, è stato effettuato un solo passo avanti (con un tasso di apprendimento α) nella direzione opposta al gradiente di T_x (calcolato sulla configurazione precedente dei pesi). Iterando questa procedura si completa la back-propagation.

Capitolo 3

Cenni storici

3.1 Neocognitron

Nel 1969, Kunihiko Fukushima introdusse la funzione di attivazione ReLU (Rectifier Linear Unit) nel contesto dell'estrazione di caratteristiche visive in reti neurali gerarchiche, sebbene solo successivamente, nel 2011, si è scoperto che essa consente un addestramento più efficace delle reti neurali più profonde, rispetto alle funzioni di attivazione ampiamente utilizzate prima del 2011, come ad esempio la sigmoide logistica. La rettificatrice è ad oggi la funzione di attivazione più popolare per le reti neurali profonde.
Dieci anni dopo, nel 1979, Kunihiko Fukushima concepì il Neocognitron, ossia una rete neurale artificiale stratificata e gerarchica. Questa architettura è stata impiegata principalmente per il riconoscimento di caratteri giapponesi scritti a mano. Si tratta di un primo esempio di CNN applicato al campo della computer vision, anche se all'epoca non fu definita in questo modo, essa è servita da fonte di ispirazione per lo sviluppo delle reti neurali convoluzionali. L'immagine 3.1 ne esemplifica il funzionamento.

La genesi delle reti neurali convoluzionali trova le sue radici in una sfida emersa nel dominio della visione artificiale: la classificazione dei numeri MNIST (modified National Institute of Standards and Technology database) e, più in generale, i processi di riconoscimento ottico dei caratteri (OCR). L'obiettivo principale dell'OCR è quello di consentire ai computer di interpretare, comprendere e trattare il testo presente in documenti fisici, consentendo così la ricerca, l'analisi e la manipolazione di contenuti testuali senza la necessità di trascrivere manualmente ogni carattere (Figura 3.2). In particolare, per quanto riguarda il dataset MNIST, la sfida consisteva nel discernere cifre scritte a mano da 0 a 9, una complessità apparentemente modesta per un essere umano, ma impegnativa per i modelli computazionali dell'epoca.

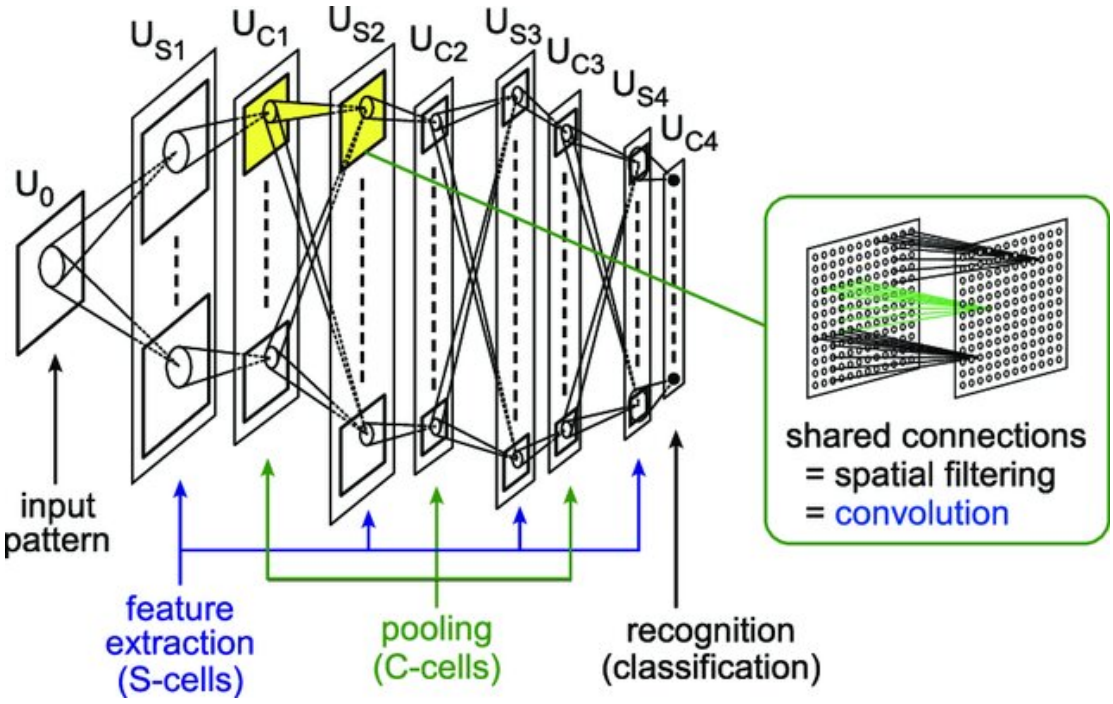

Figura 3.1: La struttura di un Necognitron, si può facilmente notare la somiglianza alle più recenti CNN

3.2 LeNet

Le reti neurali convoluzionali (CNN) sono state proposte da Yann LeCun come una soluzione innovativa ai problemi di OCR e da lì utilizzate poi in tutti i compiti di classificazione di immagini. Nella sua ricerca degli anni '90, LeCun ha introdotto una nuova architettura denominata LeNet-5, caratterizzata dall'applicazione di strati convoluzionali mirati alla comprensione di modelli gerarchici nei dati. Questa pionieristica rete neurale è stata concepita tenendo presente diverse idee chiave, tra cui:

- Campi recettivi locali: Differenziandosi dalle reti neurali tradizionali, in cui ogni neurone di uno strato è connesso a tutti quelli del layer precedente, LeNet adotta un approccio innovativo in cui ogni neurone di uno strato è connesso solo a una ristretta regione dello strato precedente. Ciò conduce a una significativa riduzione dei parametri, mitigando il rischio di overfitting e permettendo alla rete di apprendere caratteristiche locali pertinenti al compito in questione

- Condivisione dei pesi: LeNet implementa la condivisione dei pesi, una tecnica in cui ogni neurone all'interno di un sottogruppo (comunemente denominato feature map, ovvero una matrice bidimensionale di neuroni) condivide

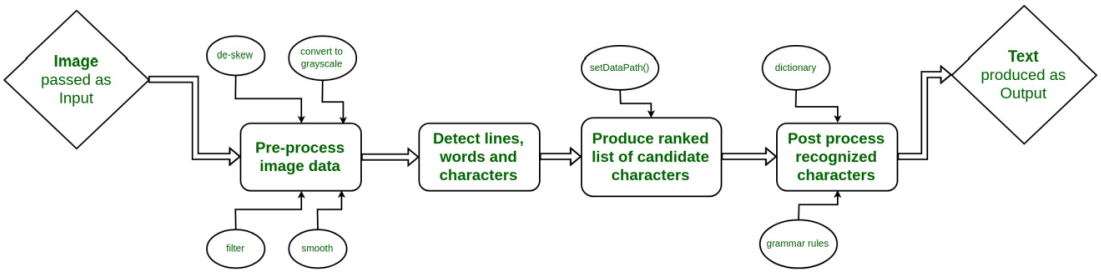

Figura 3.2: Lo schema esemplificativo di un processo OCR

lo stesso set di pesi con gli altri neuroni appartenenti alla medesima mappa. Tale condivisione dei pesi contribuisce ulteriormente a ridurre il numero complessivo di parametri, incentivando la rete a identificare caratteristiche invarianti alle piccole traslazioni dell'input.

- Sottocampionamento (o Pooling): L'architettura LeNet incorpora layer di sottocampionamento per ridurre le dimensioni delle feature map, preservando al contempo le informazioni cruciali. Questa strategia mira a diminuire il costo computazionale complessivo della rete e a prevenire potenziali problemi di overfitting.

- Strati convoluzionali: La parte fondante di una rete neurale convoluzionale risiede negli strati convoluzionali. L'impiego di questi ultimi consente alla rete di apprendere caratteristiche rilevanti indipendentemente dalla loro posizione nello spazio. Tale capacità si dimostra particolarmente rilevante nei contesti della computer vision, come nel riconoscimento di cifre scritte a mano, dove la localizzazione del carattere alfanumerico può variare.

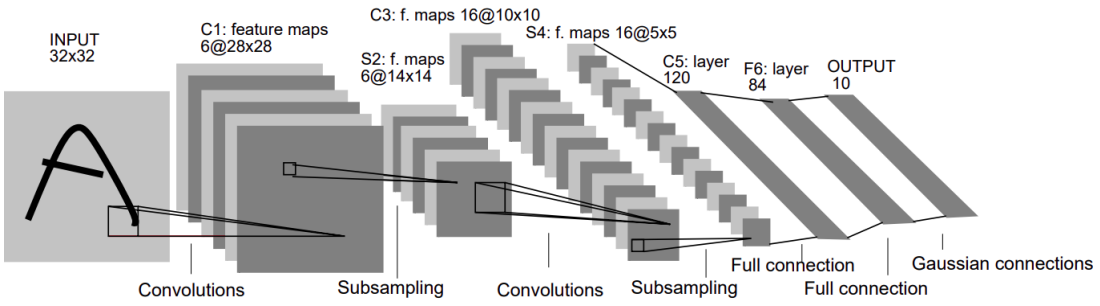

Figura 3.3: L'architettura di LeNet 5, la versione finale presentata da LeCun per l'analisi del dataset MNIST, è composta da due strati convoluzionali, due strati di sottocampionamento e due strati completamente connessi.

Nel complesso, la struttura di LeNet 5 (figura 3.3) evidenzia la potenza delle reti neurali convoluzionali nel contesto delle attività di riconoscimento delle immagini, aprendo la strada a numerosi sviluppi successivi negli algoritmi di deep learning. In particolare, lo sviluppo delle deep neural networks ha contribuito a progressi significativi in svariati ambiti, tra cui il riconoscimento di immagini, il riconoscimento vocale e l'elaborazione del linguaggio naturale: il successo di LeNet nel dataset MNIST ha catalizzato una rapida espansione che si è protratta per diverse decadi. Questo fenomeno è stato agevolato anche dallo sviluppo tecnologico in rapida crescita nel settore delle GPU (graphical computing units) e di altri hardware specializzati, oltre che dalla disponibilità sempre maggiore di dataset di dimensioni considerevoli adatti all'addestramento dei modelli, e infine, dalla diffusione di software open-source per l'addestramento automatico e la valutazione di modelli, come ad esempio PyTorch e TensorFlow. Le GPU si sono rivelate l'hardware preferito per l'implementazione di modelli neurali proprio grazie alla loro architettura, specializzata nel rendering e manipolazioni di immagini e con la capacità di svolgere molteplici operazioni computazionali parallelamente.

3.3 La sfida ImageNet

La competizione ImageNet, conosciuta come ImageNet Large Scale Visual Recognition Challenge (ILSVRC), ha costituito un palcoscenico sul quale numerosi team di ricerca hanno sviluppato algoritmi capaci di riconoscere e classificare una vasta gamma di oggetti e scene. Svoltasi annualmente dal 2010 al 2017, questa competizione ha giocato un ruolo cruciale nello sviluppo delle reti neurali convoluzionali. Prende inoltre il nome di ImageNet anche l'immenso dataset su cui i modelli sviluppati in questi anni vengono testati. Differenziandosi dal dataset MNIST, l'insieme di dati di ImageNet presenta un considerevole volume di immagini RGB, approssimativamente 1,4 milioni, distribuite in 1 000 classi distinte. Le immagini presentano una risoluzione superiore, consentendo una maggiore precisione nei dettagli. Questa diversità ha consolidato la reputazione del dataset ImageNet come un affidabile insieme di dati etichettati per una vasta gamma di compiti successivi nel campo della visione artificiale.
Nel 2012, un modello di deep learning noto come AlexNet, sviluppato attraverso l'impiego di reti neurali convoluzionali da Alex Krizhevsky, Ilya Sutskever, Geoffrey Hinton et al. presso l'Università di Toronto, ha trionfato nella competizione con un notevole margine, segnando una pietra miliare nel campo della computer vision. Le principali ragioni del successo di AlexNet includono:

- Dataset esteso: in primis, AlexNet ha avuto il vantaggio di essere addestrato sul vasto dataset ImageNet. Questo ampio insieme di dati ha consentito alla rete di apprendere una diversificata gamma di caratteristiche e modelli essenziali per il compito di classificazione delle immagini.

- Architettura profonda: AlexNet si configura come un'architettura di rete neurale profonda, composta da otto strati, di cui cinque convoluzionali e tre completamente connessi. L'utilizzo di una struttura multistrato ha reso la rete capace di catturare rappresentazioni gerarchiche complesse nelle immagini in input, fondamentali per una classificazione accurata.

- Strati convoluzionali: L'impiego di strati convoluzionali ha fornito ad AlexNet la capacità di apprendere caratteristiche in modo indipendente dalla loro posizione, un aspetto cruciale nell'analisi delle immagini. Tramite il layer di pooling, la rete ha appreso una serie di filtri convoluti in relazione all'immagine di input, generando feature map che catturano diversi aspetti dell'immagine.

- Attivazione ReLU: AlexNet ha incorporato la funzione di attivazione ReLU (rectified linear unit), accelerando notevolmente il processo di apprendimento rispetto alle architetture precedenti basate su attivazioni sigmoidali o tangenti iperboliche. ReLU ha altresì contribuito a mitigare il problema del gradiente che tende a svanire con il crescere della profondità delle reti neurali.

- Data augmentation: AlexNet ha implementato tecniche di data augmentation, quali cropping, flipping, and scaling per incrementare artificialmente le dimensioni del dataset di addestramento. Queste tecniche aumentano la dimensionalità del dataset a parità di parametri del modello, insegnando alla rete a gestire trasformazioni che non impattano sulla classificazione. Ciò ha contribuito a mitigare l'overfitting, migliorando le prestazioni di generalizzazione della rete.

- Dropout regolarization: AlexNet ha adottato una tecnica di ottimizzazione denominata Dropout regolarization, che opera durante la fase di addestramento del modello, durante la quale casualmente "elimina" un determinato numero di unità in ciascuno strato della rete con una probabilità specificata, solitamente compresa tra il 20% e il 5%. La topologia della rete rimane così la stessa ma va a ridursi il numero di connessioni attive. Questo processo simula l'effetto di addestrare diverse reti neurali con un numero ridotto di connessioni invece di una singola rete grande. Di conseguenza, il modello deve imparare rappresentazioni robuste e generali dei dati, riducendo così la tendenza all'overfitting. Tale strategia ha altresì contribuito a migliorare le prestazioni di generalizzazione della rete.

3.4 ResNet

Nonostante le qualità mostrate dalle reti precedenti, l'addestramento di reti particolarmente profonde ha incontrato un ostacolo significativo: il problema della scomparsa del gradiente. Con l'aumentare della profondità della rete, i gradienti calcolati durante la retropropagazione diventavano spesso estremamente piccoli, ostacolando l'efficace apprendimento della rete.

ResNet, abbreviazione di Residual Network, sviluppato da Microsoft Research,

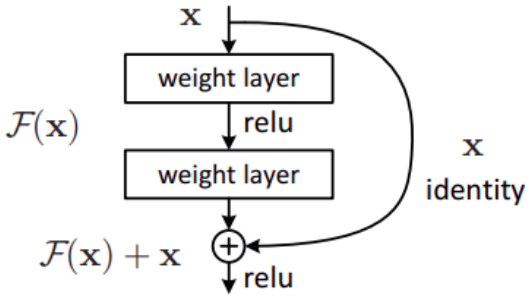

Figura 3.4: Struttura di un blocco di ResNet

ha risolto questa sfida introducendo dei "collegamenti di salto" o "shortcut" (immagine 3.4), che consentono ai gradienti di retropropagare le correzioni dei pesi direttamente ai livelli precedenti. Questa architettura ha permesso l'addestramento di reti notevolmente più profonde rispetto al passato; ad esempio, il team di ricerca di Microsoft ha presentato un modello con 152 strati durante la competizione ImageNet del 2015. Ciò ha rappresentato un progresso significativo, permettendo l'addestramento efficiente di reti con profondità variabile e ottenendo prestazioni superiori rispetto alle controparti meno profonde.

3.5 EfficientNet

EfficientNet, una rete neurale convoluzionale (CNN) sviluppata per ottimizzare l'utilizzo delle risorse computazionali, è stata pubblicata per la prima volta nel 2019. La ricerca è stata condotta da Mingxing Tan e Quoc V. Le di Google Brain e la pubblicazione è intitolata "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks", ed è stata presentata all'annuale International Conference on Machine Learning (ICML) . La rete qui menzionata è stata concepita con l'intento di ottenere un equilibrato rapporto tra precisione e complessità computazionale, consentendo così la creazione di modelli che risultino efficienti in termini di risorse impiegate. Tra le principali innovazioni introdotte da EfficientNet spiccano:

- Compound Scaling: EfficientNet adotta un approccio di "compound scaling" che bilancia uniformemente le dimensioni della rete su tre fondamentali aspetti: larghezza (width), profondità (depth) e risoluzione (resolution) dell'input. Tale metodologia permette di scalare la rete in maniera equilibrata su tutti e tre i fronti, garantendo prestazioni superiori con un utilizzo efficiente delle risorse computazionali.

- Efficient Building Blocks: EBB è un modulo che comprende più strati di convoluzione e altre operazioni, progettato per massimizzare l'efficienza del modello. Questi blocchi sono stati progettati in modo da ottenere una buona rappresentazione delle caratteristiche dell'input riducendo al contempo il numero di parametri e le operazioni computazionali richieste.

- La progettazione di EfficientNet si avvale di un nuovo processo chiamato AutoML (Automated Machine Learning) volto a ricercare automaticamente la migliore combinazione di iperparametri, ovvero quei parametri il cui valore non viene appreso durante il processo di addestramento del modello stesso, ma devono essere stabiliti prima dell'avvio dell'addestramento e possono influenzare significativamente le prestazioni e il comportamento della rete. Questi iperparametri sono diversi dai parametri "ordinari" del modello, come i pesi delle connessioni neurali, che invece vengono ottimizzati durante l'addestramento. Questa innovativa procedura ha consentito di sviluppare modelli efficienti in termini di risorse e dotati di prestazioni competitive su un ampio spettro di compiti di visione artificiale. In particolare, l'ottimizzazione della stima degli iperparametri è un argomento attualmente discusso. Solitamente oggi si ricorre ancora a sistemi analoghi ad AutoML, tecniche di meta-learning che cercano di apprendere gli iperparametri migliori attraverso ripetuti esperimenti di learning.

Grazie a queste innovazioni, EfficientNet ha assunto un ruolo preminente nel campo del deep learning, fornendo modelli di reti neurali efficienti e ad alte prestazioni per una variegata gamma di compiti di visione artificiale e stabilendo uno Standard nell'ottimizzazione di essi nei confronti dell'efficienza, come il nome lascia intendere.

3.6 L'avvento dei Trasformer

Le reti neurali convoluzionali, grazie alla loro capacità intrinseca di catturare gerarchie spaziali, hanno conosciuto ampio utilizzo e successo. Le loro architetture sin dalla pionieristica LeNet hanno subito un costante processo evolutivo per migliorare l'estrazione delle caratteristiche, per gestire reti più profonde e per eseguire calcoli efficienti con un numero minore di dati. Tuttavia, l'introduzione dei modelli Transformer, inizialmente presentati nel celebre paper "Attention is All You Need" di Vaswani et al., ha posto una sfida alla supremazia delle reti neurali convoluzionali, sia nel campo della Computer Vision che in altri settori, in particolare essi detengono tuttora l'egemonia nei campi dello Speech Recognition e Text Generation. In particolare, il celebre chatbot ChatGPT è basato su un modello Transformer, nello specifico sulla versione GPT (Generative Pre-trained Transformer) sviluppata da OpenAI. A differenza delle reti neurali convoluzionali, i Transformer non seguono un processo sequenziale di elaborazione dati, infatti sfruttano un meccanismo chiamato autoattenzione che permette loro di considerare l'intera sequenza simultaneamente. Ciò conferisce ai Transformer una notevole efficacia nel modellare dipendenze a lungo raggio nei dati, una qualità particolarmente preziosa nei compiti come la traduzione di lingue e la sintesi di testi. Inoltre, essi hanno gradualmente invaso anche il campo della computer vision, superando le CNN in vari contesti.
La ragione per cui le reti neurali convoluzionali hanno predominato nella computer vision risiede nella limitazione del numero di pixel adiacenti che possono influenzare un altro pixel attraverso piccoli filtri convoluzionali, a differenza delle altre ANN che considerano tutti i pixel come vicini. I Transformer rendono apprendibile questo processo di considerazione dei vicini utilizzando ANN più classiche come i multilayer perceptrons (MLP), conferendo alla rete la capacità di considerare ogni pixel come un vicino e incorporando il meccanismo di attenzione. Questo è ulteriormente arricchito da MLP che permettono alla rete di apprendere quali pixel siano più rilevanti e quali meno.

Tuttavia, la sfida tra le reti neurali convoluzionali e i Transformer è lontana dall'essere conclusa: un recente lavoro di Zhuang Liu et al. ha presentato un miglioramento dell'architettura di una rete neurale convoluzionale ResNet-50, superando in prestazioni l'architettura Transformer di ultima generazione, Swin-T. Le principali modifiche apportate a ResNet-50 per il miglioramento delle prestazioni includono:

- Tecniche di addestramento: Gli autori hanno sottolineato che dopo la pubblicazione di ResNet, sono state inventate nuove tecniche di addestramento e di ottimizzazione. Pertanto, hanno modificato il processo di addestramento di ResNet-50 utilizzando gli avanzamenti recenti nel campo del data augmentation applicando le tecniche di ultima generazione.

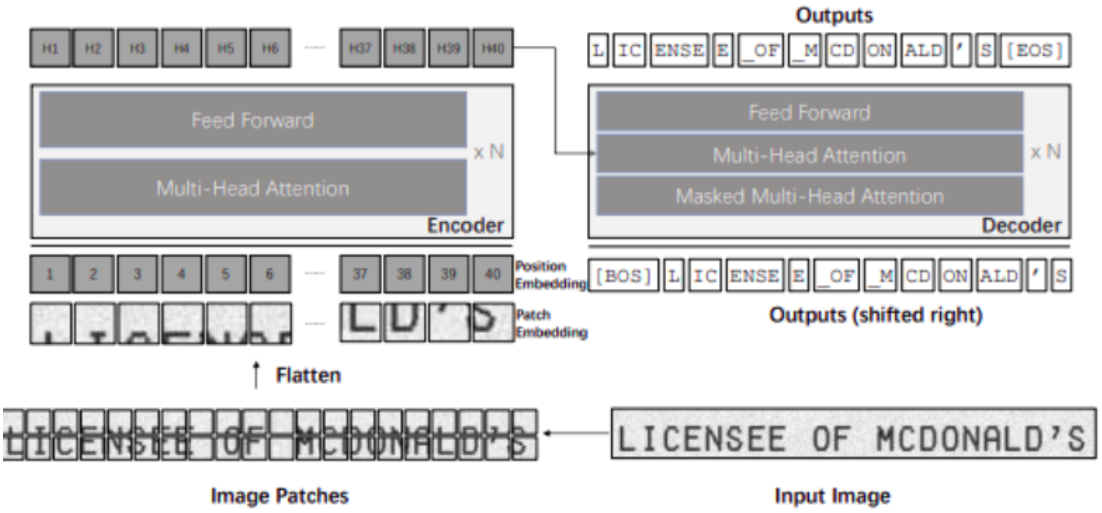

Figura 3.5: Attraverso l'uso di un processo di trasformazione, in cui ciascun pixel dell'immagine viene rappresentato come un vettore. Successivamente, l'architettura trasformer applica meccanismi di attenzione per catturare le relazioni spaziali tra i pixel, consentendo al modello di elaborare e comprendere l'informazione visiva.

- Funzioni di attivazione: Gli autori hanno deciso di non scegliere la funzione di attivazione ReLU, prestando attenzione al fatto che nel dominio dell'elaborazione del linguaggio naturale, dove le architetture dei Transformer sono così di successo, le ReLU non sono utilizzate tanto quanto le funzioni Sigmoid e TanH. Hanno sostituito la ReLU con la Gaussian Error Linear Unit (GELU), una versione più fluida delle ReLU. Hanno anche ridotto il numero di volte in cui le funzioni di attivazione sono state utilizzate nella rete, dimostrando come ciò sia vantaggioso.

- Dimensioni del filtro più grandi: Gli autori sottolineano il fatto che i livelli dell'architettura Transformer hanno campi recettivi più ampi grazie al loro meccanismo di autoattenzione. A differenza di ciò, nelle reti neurali convoluzionali, il valore tipico delle dimensioni del filtro storicamente è sempre stato 3x3, e quasi tutti i modelli successivi non hanno utilizzato filtri più grandi di quelli. In questo caso, gli autori hanno dimostrato che l'utilizzo di kernel 7x7 nelle convoluzioni separabili in profondità non solo aumenta le prestazioni, ma ha anche un impatto trascurabile sulla complessità computazionale espressa in FLOPs, poiché le componenti in profondità di queste convoluzioni richiedono significativamente meno pesi.

Ricordiamo che FLOPs è un'abbreviazione di "floating point operations per second" (operazioni in virgola mobile al secondo). Rappresenta il numero di operazioni di punto mobile eseguite da un sistema di calcolo in un secondo. Le FLOPs

sono spesso utilizzate come misura della potenza di calcolo di un sistema. Nei contesti di deep learning, le FLOPs solitamente vengono scelte per quantificare il carico computazionale richiesto per eseguire un'operazione su un modello, come una moltiplicazione di matrici durante la fase di inferenza o di addestramento. Questo aiuta a stimare la velocità con cui un modello può essere eseguito su una determinata piattaforma hardware e a confrontare l'efficienza computazionale di diversi modelli o architetture.

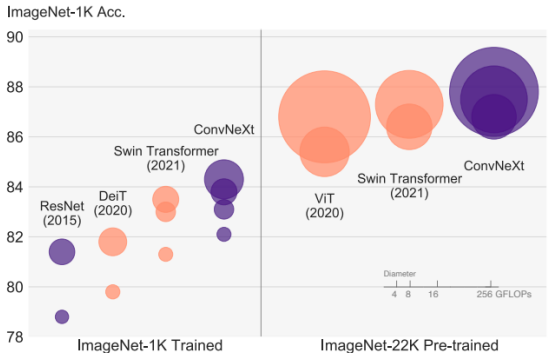

Figura 3.6: I risultati della classificazione ImageNet-1K ottenuti attraverso l'analisi comparativa tra ConvNets e Transformers vision. Ogni bolla rappresenta una variante di un modello all'interno delle rispettive famiglie, con la dimensione della bolla correlata ai FLOPs associati al modello.

È importante notare che i risultati relativi a ResNet e ViT mostrati nell'immagine 3.6 sono stati ottenuti mediante l'implementazione di procedure di addestramento ottimizzate rispetto agli approcci descritti nei rispettivi paper originali. L'analisi condotta dagli autori rivela che, nonostante la maggiore complessità dei Transformers vision, i modelli CNN standard possono raggiungere un livello di scalabilità paragonabile, mantenendo nel contempo un design notevolmente più semplice. Nel contesto dell'analisi della precisione di un modello di machine learning, emerge anche una distinzione significativa tra i concetti di "pre-trained" e "trained", che riguarda principalmente il processo di addestramento e l'utilizzo del modello.

3.6.1 Modelli Trained e Pretrained

Un modello pre-addestrato, o pre-trained, è un modello che è stato precedentemente allenato su un vasto e generalizzato dataset, spesso attraverso l'impiego di considerevoli risorse computazionali. Questi modelli vengono solitamente addestrati per affrontare un'ampia gamma di problemi, come il riconoscimento di immagini o la classificazione del linguaggio naturale, coprendo un vasto spettro di categorie. Dopo il periodo di addestramento, il modello viene reso disponibile pubblicamente e può essere utilizzato come punto di partenza per risolvere

problemi specifici. Ciò avviene tramite un processo noto come "fine-tuning", che consiste nell'adattare il modello ai nuovi dati o al nuovo dominio di interesse. Nella valutazione della precisione di un modello pre-addestrato su un nuovo problema, è comune considerare le performance del modello prima e dopo il fine-tuning. Un modello addestrato, o trained model, invece è un modello che è stato specificamente allenato su un dataset di dati specifici per risolvere un particolare problema. Questo processo di addestramento coinvolge l'ottimizzazione dei parametri del modello per massimizzare le sue performance sul compito di interesse. La precisione di un modello addestrato si riferisce alla sua capacità di generalizzare correttamente su dati non osservati precedentemente, dopo il periodo di addestramento. Nella valutazione della precisione di un modello addestrato, solitamente si considerano le performance su un set di dati di test separato, che non è stato utilizzato durante l'addestramento, al fine di valutare la sua capacità di generalizzazione.

3.7 I foundation models

La prossima fase nel campo dell'Intelligenza Artificiale mira a superare i modelli specializzati in singoli compiti che hanno finora dominato il panorama dell'IA. Lo sforzo collettivo della comunità scientifica sembra indirizzato alll'impiego di modelli addestrati su vasti insiemi di dati non etichettati, capaci di essere applicati a diversi compiti con minimo fine-tuning, ovvero addestramento al compito specifico. Questi modelli sono noti come Foundation Models, un termine introdotto per la prima volta dallo Stanford Institute for Human-Centered Artificial Intelligence. I Foundation Models rappresentano ampie implementazioni su larga scala che fungono da punto di partenza primario per affrontare una vasta gamma di compiti di machine learning, tra cui il riconoscimento di immagini e di linguaggio naturale. Questi modelli vengono sottoposti a un processo di preaddestramento su dataset estesi, come ImageNet per le immagini e Common Crawl per il testo, attraverso l'utilizzo di notevoli risorse computazionali e dati annotati. Una volta che il preaddestramento è completato, i modelli fondamentali offrono notevoli vantaggi quando vengono impiegati per risolvere nuovi compiti. Tale procedura, nota come trasferimento di conoscenza o transfer learning, consente di evitare l'impiego di tempo e risorse necessarie per addestrare un modello da zero. Piuttosto, si parte da un modello preaddestrato e si adatta ai nuovi dati attraverso il fine-tuning. Durante il fine-tuning, il modello preaddestrato subisce aggiornamenti per adeguarsi al compito specifico. Ciò può comportare la sostituzione di alcuni strati del modello con nuovi strati adatti al nuovo compito o semplicemente l'aggiornamento dei pesi degli strati esistenti in base ai nuovi dati. I foundation models riescono a generalizzare in modo più efficace e richiedono una quantità inferiore di dati etichettati per il fine-tuning rispetto all'addestramento da zero.

Ecco una lista dei foundation models più conosciuti attualmente utilizzati nella computer vision.

- YOLOv8
- MobileNetV2
- MobileNetV2
- OWL-ViT

Conclusioni

Le reti neurali convoluzionali hanno rivoluzionato il campo della computer vision, consentendo un notevole progresso nelle applicazioni di riconoscimento di immagini e pattern. Nonostante il processo di sviluppo e ottimizzazione intrapreso nelle ultime due decadi, non è possibile stabilire con certezza se esse siano ancora attualmente lo strumento più efficace in questo campo, essendo tuttora in competizione con i Trasformer. Negli ultimi anni, l'evoluzione dei "foundation models" ha aggiunto un nuovo capitolo significativo. Questi modelli, in particolare i transformer-based, hanno dimostrato di raggiungere prestazioni eccezionali in una varietà di compiti di visione artificiale e linguaggio naturale. Uno dei vantaggi più rilevanti dei foundation models è la loro capacità di raggiungere alte prestazioni anche con una quantità limitata di dati di addestramento, essendo stati pre-addestrati senza uno scopo preciso. Questo è un cambiamento significativo rispetto alla necessità precedente di grandi dataset per ottenere risultati competitivi. Inoltre, grazie alla loro struttura altamente ottimizzata, questi modelli possono essere eseguiti anche su hardware relativamente modesti, rendendo l'implementazione pratica per una vasta gamma di dispositivi e applicazioni. Tuttavia, è necessario che venga mantenuta una filosofia open-source per questi foundation models per garantire l'accesso alla maggior parte degli sviluppatori. La proprietà esclusiva di questi modelli potrebbe limitare l'accesso e l'innovazione, creando barriere all'entrata nel settore e limitando le opportunità di ricerca ed ottimizzazione di queste tecnologie ancora in forte sviluppo.

Bibliografia

[1] Albawi, Saad et al. *Understanding of a Convolutional Neural Network*. 2017.

[2] Bhatt, Dulari et al. *CNN Variants for Computer Vision: History, Architecture, Application, Challenges and Future Scope*. 2021.

[3] Bianchi, Alessandro, Vendra Moreno Raimondo. *Optimization of Convolutional Neural Networks: Transfer Learning for Robustness to Image Distortion through Selective Filter-Level Fine-Tuning*. 2019.

[4] Bommasan, Rishi et al. *On the Opportunities and Risks of Foundation Models*. 2022.

[5] Dehghani, Mostafa et al. *Scaling Vision Transformers to 22 Billion Parameters*. 2023.

[6] Fukushima, Kunihiko. *Neural network model for a mechanism of pattern recognition unaffected by shift in position — Neocognitron — Trans*. 1979.

[7] Goodfellow, Ian et al. *Deep Learning Book*. s.d.

[8] He, Kaiming et al. *Deep Residual Learning for Image Recognition*. 2015.

[9] Krizhevsky, Alex et al. *ImageNet Classification with Deep Convolutional Neural Networks*. 2012.

[10] LeCun, Yan et al. *Gradient-Based Learning Applied to Document Recognition*. 1998.

[11] Liu, Zhuang et al. *A ConvNet for the 2020s*. 2022.

[12] Lomonaco, Vincenzo. *Deep Learning for Computer Vision: A comparison between Convolutional Neural Networks and Hierarchical Temporal Memories on object recognition tasks*. 2015.

[13] Maurício, José et al. *Comparing Vision Transformers and Convolutional Neural Networks for Image Classification: A Literature Review*. 2023.

[14] Maiani, Fabio. *Applicazioni e limiti della classificazione di immagini con reti neurali convoluzionali in dispositivi mobili*. 2016.

[15] McKinney, O'reilly. *Python for Data Analysis*. 2012.

[16] O'Shea, Keiron, Nash, Ryan. *An Introduction to Convolutional Neural Networks*. 2015.

[17] Prince, Simon J.D. *Understanding Deep Learning*. 2024.

[18] Tan, Mingxing, Le Quoc. *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. 2019.

[19] Vaswani, Ashish et al. *Attention Is All You Need*. 2017.

[20] IBM. *What Are Foundation Models*. `https://research.ibm.com/blog/what-are-foundation-models`.

[21] MDPI. *A Guide to Convolutional Neural Networks*. `https://www.mdpi.com/2076-3417/13/9/5521`

[22] Labelbox. *6 Cutting-Edge Foundation Models for Computer Vision and How to Use Them*. `https://www.labelbox.com/blog/6-cutting-edge-foundation-models-for-computer-vision-and-how-to-use`

[23] Labelbox. *6 Cutting-Edge Foundation Models for Computer Vision and How to Use Them*. `https://labelbox.com/blog/6-cutting-edge-foundation-models-for-computer-vision-and-how-to-use-them`

‘‘‘